

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему

**“Програмне забезпечення системи кібербезпеки на основі
реалізації методів соціального інжинірингу через адресу e-mail”**

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-1
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Вовк Д.О.
«___» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 125 "Кібербезпека"
Освітньо-професійна (освітньо-наукова) програма "Кібербезпека"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 27 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Вовку Данилі Олеговичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 134-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту 23.05.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи кібербезпеки 1 аркуш

Функціональна схема системи кібербезпеки 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 2 аркуша

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Вовк Д.О.
(прізвище та ініціали)

АНОТАЦІЯ

Вовк Д.О. Програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

Метою розробки є програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

Результат роботи – програмна реалізація системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, соціальний інжиніринг

ABSTRACT

Vovk D.O. Software for a cybersecurity system based on the implementation of social engineering methods via an e-mail address. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a cybersecurity system based on the implementation of social engineering methods via an e-mail address.

The purpose of the development is to develop software for a cybersecurity system based on the implementation of social engineering methods via an e-mail address.

The result of the work is a software implementation of a cybersecurity system based on the implementation of social engineering methods via an e-mail address.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: cybersecurity, social engineering

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи.....	5
1.2 Область застосування.....	5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	7
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування.....	14
2.3 Розгорнута постановка завдання	16
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	18
3.1 Опис функціонування системи	18
3.2 Розробка структурної схеми.....	22
3.3 Розробка функціональної схеми	27
3.4 Розробка діаграми процесів.....	28
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	30
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	30
4.2 Захист розробленого програмного забезпечення.....	47
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	52
6 ОСНОВНІ ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	62

					ВКРБ-125.26.0004.00.00.ПЗ			
Вим.	Арк.	№ докум.	Підп.	Дата	<i>Програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail</i>	Літ.	Аркуш	Аркушів
<i>Розроб.</i>	<i>Вовк Д.О.</i>					Б	1	70
<i>Перев.</i>	<i>Смірнов С.А.</i>					ЦНТУ КБ-22-1		
<i>Н.контр.</i>	<i>Коваленко А.С.</i>							
<i>Затв.</i>	<i>Смірнов О.А.</i>							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DHCP	протокол, що використовується для динамічного розподілу IP-адрес.
DNS	розподілена служба Інтернет, використовувана для зіставлення логічних (доменних) імен і IP-адрес. DNS використовується для забезпечення можливості роботи зі зрозумілими й іменами, що легко запам'ятовуються, замість IP-адрес у числовому форматі
ICMP	протокол
IDS	система виявлення атак
IP	адресний протокол
LAN	локальна мережа
NAT	трансляція IP-адрес із одного адресного простору в IP-адреси іншого адресного простору
Proxy	програма-посередник, що транслює запити різних протоколів із локальної мережі в зовнішню мережу
TCP	протокол обміну даними на транспортному рівні
UDP	протокол
URL	уніфікований показник інформаційного ресурсу (стандартизований рядок символів, що вказує місцезнаходження документа в мережі Інтернет)
ЕЦП	електронний цифровий підпис
КД	ключова дискета
НСД	несанкціонований доступ
ОС	операційна система
ПЗ	програмне забезпечення
ПК	персональний комп'ютер

ВСТУП

Актуальність теми. «Щомісячне моделювання фішингу» не є більш ефективним, ніж ваша стратегія управління ризиками для людини. У 2026 році кіберзлочинці, які здійснюють фішингові атаки, не діють за графіком і, звичайно ж, не використовують статичні шаблони фішингу. Ми вступили в епоху агентних загроз, де атакуючі агенти зі штучним інтелектом вилучають дані ваших співробітників, створюють гіперперсоналізовані приманки та виконують кампанії зі швидкістю машини.

Якщо ваша оборонна стратегія спирається на ручні електронні таблиці або застарілі інструменти, які вимагають годин роботи адміністратора для запуску однієї кампанії, ви вже відстаєте. Сучасним стандартом є автоматизоване моделювання фішингу. Це платформи «налаштував і забув», які використовують штучний інтелект для постійного тестування, навчання та щеплення ваших співробітників від атак, які виглядають і відчуються реальними, оскільки в багатьох випадках вони генеруються тими ж моделями штучного інтелекту, що й зловмисники.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

- Огляд існуючих систем на основі реалізації методів соціального інжинірингу через адресу e-mail.
- Дослідження системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.
- Програмна реалізація системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі на основі реалізації методів соціального інжинірингу через адресу e-mail.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2026

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Симуляції фішингу – це контрольовані вправи, які організації використовують для навчання персоналу небезпеці оманливих електронних листів. Ці програми імітують реальні кібератаки, надсилаючи співробітникам фальшиві спроби фішингу. Мета полягає не в тому, щоб обдурити людей заради цього, а в тому, щоб створити сильніший людський брандмауер.

Як працюють симуляції фішингу?

Команди безпеки або постачальники спеціалізованого програмного забезпечення створюють повідомлення, які виглядають як законні запити від банків, IT-відділів або відділів кадрів. Вони часто містять типові тривожні сигнали, такі як:

- Термінова мова, що вимагає негайних дій.
- Підозрілі посилання, що ведуть на фальшиві портали входу.
- Неочікувані вкладення, що містять нешкідливі файли відстеження.

Коли працівник взаємодіє із змодельованою загрозою, його зазвичай перенаправляють до короткого навчального модуля. Це забезпечує негайний «момент навчання», який пояснює, що працівник пропустив і як виявляти подібні загрози в майбутньому.

1.2 Область застосування

Перш ніж перераховувати інструменти, важливо зрозуміти перехід від «автоматизованого» до «агентського».

- Застаріла автоматизація (2020-ті): Ви вибираєте шаблон, завантажуєте CSV-файл користувачів, плануєте випуск і переглядаєте статичний звіт.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

– Agentic Automation (2026): Програмне забезпечення діє як автономний оператор. Воно знаходить ваших співробітників із загальнодоступних джерел, створює унікальні корисні навантаження за допомогою генеративного штучного інтелекту, надсилає їх на основі поведінки користувачів (наприклад, коли вони найімовірніше перевірятимуть електронну пошту) та забезпечує миттєве мікронавчання, коли користувач дає збій.

Вам потрібен інструмент, який діє як постійний хакер, а не як машина для опитувань.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Агент симуляції фішингу від Hunto AI

Найкраще підходить для: організацій, які прагнуть повністю автономного управління ризиками з урахуванням людського фактору «без дотику».

Hunto – це не просто інструмент симуляції; це агентна платформа кібербезпеки. Її агент симуляції фішингу розроблений для повного зняття адміністративного навантаження з команди безпеки, водночас забезпечуючи найреалістичніші симуляції фішингу електронної пошти на ринку.

Основні характеристики:

– Генеративні корисні навантаження III: Hunto не покладається на статичну бібліотеку. Агент сканує цифровий слід вашої організації (новини, LinkedIn, технологічний стек), щоб генерувати фішингові атаки з урахуванням контексту, які імітують складний фішинг. Агент автоматично сканує діяльність вашої організації, співробітників, профілі співробітників у соціальних мережах та пропонує персоналізовані атаки замість шаблонів.

– Автономний життєвий цикл: Від розвідки до звітності, весь цикл автоматизовано. Агент визначає користувачів з високим ризиком і динамічно коригує рівень складності. Якщо користувач досвідчений, агент докладає більше зусиль. Якщо користувач має труднощі, агент надає коригувальне навчання.

– Номер людського ризику (HRN): Hunto надає уніфікований HRN, який оновлюється в режимі реального часу на основі результатів моделювання та реальної поведінки у сфері безпеки, надаючи Раді чіткий показник покращення.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

– Зовнішня інтеграція: інтегрується з Hunto Autopilot для співвіднесення результатів фішингу з іншими ризиками, такими як викриття в даркнеті або видання себе за бренд.

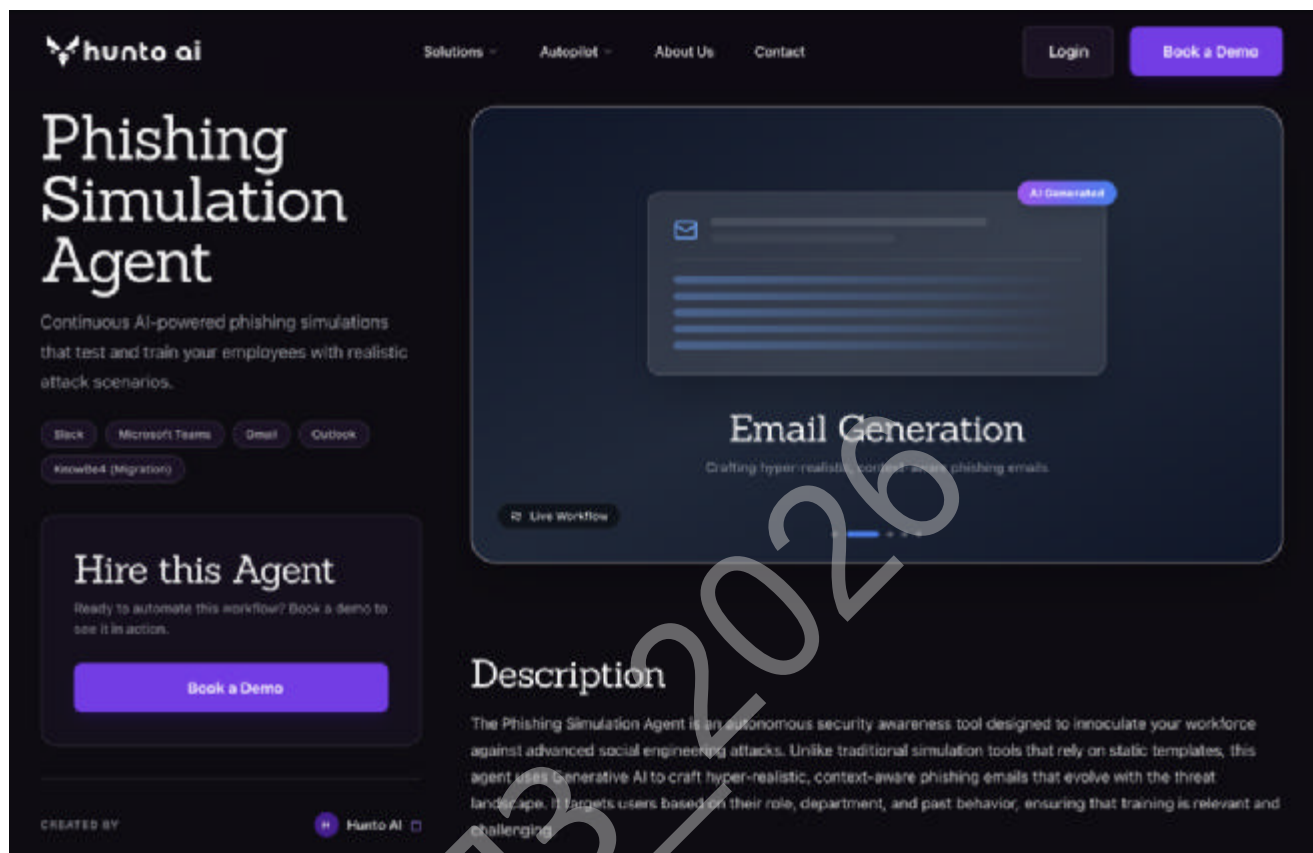


Рисунок 2.1 – Інтерфейс користувача Hunto AI

Нохhunt

Найкраще підходить для: Зміни поведінки за допомогою гейміфікації.

Нохhunt зайняла свою нішу, зосередившись на досвіді співробітників. Замість того, щоб обманювати користувачів, компанія позиціонує безпеку як гру. Їхня платформа «Управління людськими ризиками» використовує механізм на базі штучного інтелекту для створення персоналізованих симуляцій, які є власними для робочого процесу користувача.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

Чому це працює:

- Гейміфікована звітність: Користувачі отримують винагороду за повідомлення про загрози, створюючи позитивний зворотний зв'язок.
- Адаптивне навчання: Платформа аналізує поведінку користувачів, щоб створювати симуляції на межі їхнього рівня кваліфікації, гарантуючи, що вони стикаються з труднощами, але не розчаровані.
- Глобальні операції: Потужна підтримка багатомовного розгортання робить його улюбленим для глобальних підприємств.

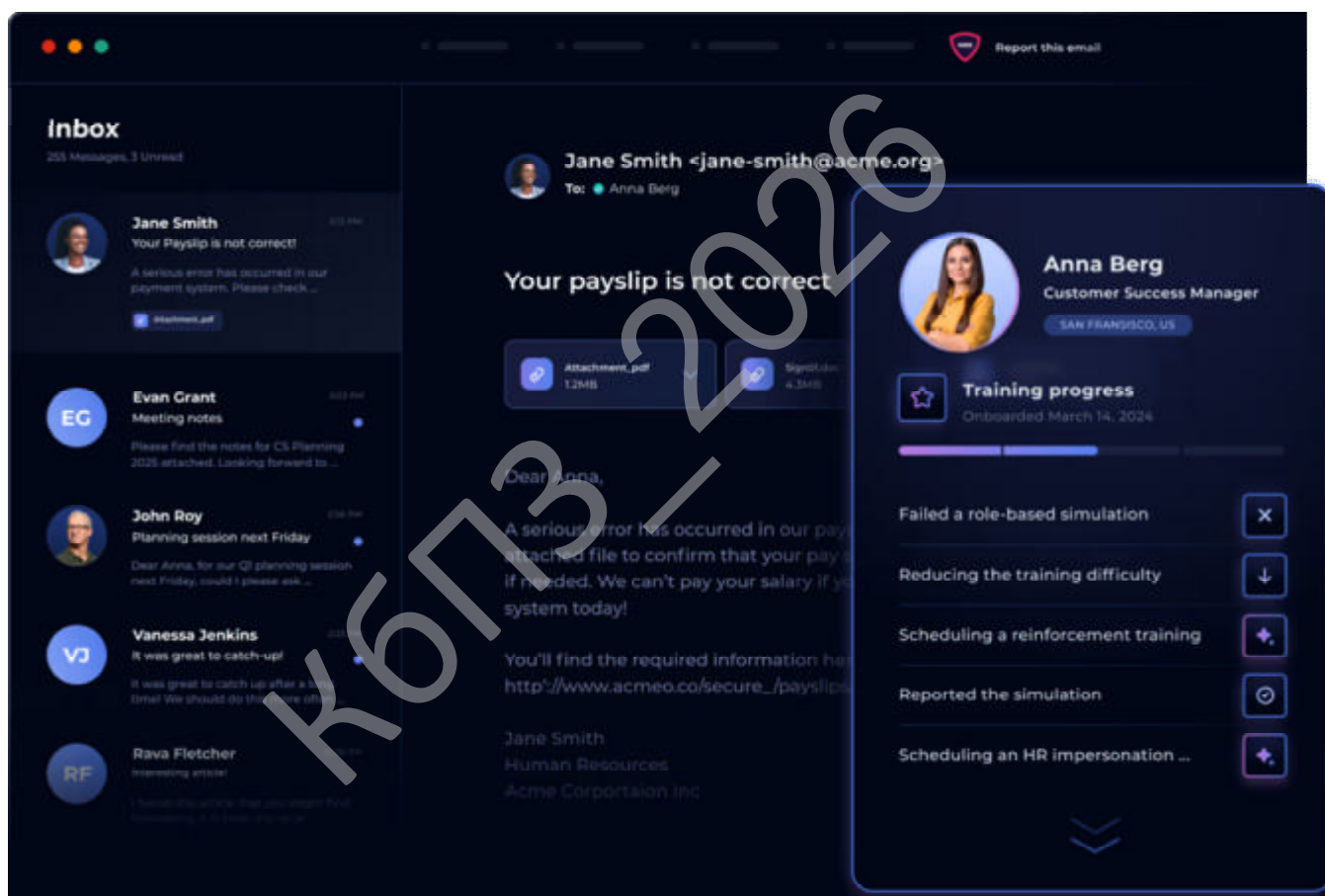


Рисунок 2.2 – Інтерфейс користувача Noxhunt

Phished.io

Найкраще для: Автоматизованої генерації контенту без участі сторонніх осіб.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

Phished.io була однією з перших компаній, що почали використовувати контент на основі штучного інтелекту. Їхня платформа робить акцент на «нульовому ручному втручанні». Штучний інтелект щодня аналізує тенденції загроз, щоб створювати нові шаблони, що відображають активні кампанії в реальних умовах.

Чому це працює:

- Розумне планування: воно визначає оптимальний час для надсилання електронного листа певному користувачеві, щоб максимізувати ймовірність взаємодії.

- Безкоштовний контрольний список: Немає складних інструкцій, яких потрібно дотримуватися; штучний інтелект керує стратегією кампанії на основі визначеного вами профілю ризику.

KnowBe4 (HRM+)

Найкраще для: Різноманітність контенту та відповідність вимогам важких галузей.

Як лідер ринку, KnowBe4 пропонує найглибшу бібліотеку навчального контенту в галузі. Їхня вдосконалена платформа HRM+ намагається подолати розрив між їхніми застарілими навчальними модулями та сучасними поведінковими даними.

Чому це працює:

- Величезна бібліотека: тисячі шаблонів та навчальних відео, включаючи високоякісні серіали, які розважають користувачів.

- Розумні групи: Дозволяє детально вибирати групи користувачів на основі поведінки, місцезнаходження або ролі.

- Розпізнаваність бренду: Безпечний вибір для аудиторів та відповідальних за дотримання вимог, які шукають відомі бренди.

Обізнаність щодо безпеки Proofpoint

Найкраще підходить для: Симуляцій на основі розвідки загроз. Proofpoint використовує своє домінування на ринку шлюзів електронної пошти (SEG) для

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

підживлення своїх симуляцій. Оскільки вони бачать величезний обсяг світового електронного трафіку, вони можуть миттєво перетворити реальні атаки, заблоковані їхнім шлюзом, на симуляції.

Чому це працює:

- VAP (люди, що дуже зазнають нападів): Визначає та спрямовує навчання на конкретних осіб у вашій компанії, які є мішенню реальних зловмисників.

- У центрі уваги атаки: Перетворює трендові глобальні загрози на навчальні модулі майже в режимі реального часу.

Cofense (раніше PhishMe)

Найкраще підходить для: Інтеграції симуляцій з SOC.

Cofense зосереджується на «підготовці», а не просто на навчанні. Їхня унікальна ціннісна пропозиція полягає в інтеграції симуляції та Центру операцій безпеки (SOC). Вони розглядають співробітників як людських датчиків, які можуть повідомляти про активні загрози.

Чому це працює:

- PDC (Центр захисту від фішингу): Перевіряє електронні листи, про які повідомляли користувачі, щоб відокремити реальні загрози від симуляцій.

- Реагування на інциденти: безпосередньо передає симуляції фішингу в робочий процес реагування на інциденти, навчаючи команду SOC разом із загальним персоналом.

Ironscale

Найкраще підходить для: Інтегрована безпека та симуляція електронної пошти. Ironscale поєднує захист та навчання в одній платформі. Вона використовує штучний інтелект для виявлення аномалій у вхідних повідомленнях та використовує той самий інтелект для створення симуляцій.

Чому це працює:

- Themis Co Pilot : помічник на основі штучного інтелекту, який допомагає класифікувати загрози та пропонує сценарії моделювання.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

– Безперебійний зворотний зв'язок: Користувачі повідомляють про фішинг безпосередньо в Outlook/Gmail, і платформа миттєво аналізує повідомлення.

Arctic Wolf: керована обізнаність про безпеку

Найкраще підходить для: організацій, які хочуть передати весь процес на аутсорсинг.

Arctic Wolf пропонує модель «Консьерж-безпека». Замість інструменту вони надають вам послугу. Їхня команда керує кампаніями, аналізує дані та надає звіти.

Чому це працює:

– Мікронавчання: зосереджено на коротких, легко засвоюваних сесіях контенту (часто <3 хвилин) для підтримки уваги.

– Нульовий адміністрування: ідеально підходить для економних ІТ-команд, які не мають достатньої пропускну здатності для керування інструментом моделювання.

Залучення Mimecast

Найкраще для: Високоякісного, гумористичного навчального контенту.

Mimecast (раніше Ataata) відомий тим, що розуміє, що нудні тренінги ігноруються. Їхній контент використовує гумор та відео в стилі ситкомів, щоб зацікавити користувачів, прагнучи зменшити фактор «закочування очей» під час тренінгів з безпеки.

Чому це працює:

– Дивабельність: Контент, який співробітники фактично обговорюють біля кулера з водою.

– CyberGraph: Створює графік взаємозв'язку між користувачами та зовнішніми відправниками для налаштування оцінок ризику.

Sophos Phish Threat

Найкраще підходить для: існуючих користувачів Sophos Central. Якщо ви вже користуєтеся брандмауерами або захистом кінцевих точок в екосистемі

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

Sophos, Phish Threat – логічний вибір. Керування ним здійснюється з тієї ж панелі інструментів «Sophos Central», що дозволяє створювати консолідовану звітність.

Чому це працює:

- Уніфікована консоль: керуйте навчанням з безпеки кінцевих точок, мережі та електронної пошти в одній вкладці.
- Синхронізована безпека: Може співвідносити збій симуляції користувача зі станом справності його кінцевої точки.

Infosec IQ

Найкраще підходить для: Глибокої кастомізації та навчання на основі ролей.

Infosec IQ пропонує надійні функції для організацій, які хочуть адаптувати навчання до певних ролей (наприклад, розробники чи HR). Він забезпечує детальний контроль над шаблонами симуляцій та навчальними цільовими сторінками.

Чому це працює:

- Оцінка робочої сили: Вимірює «культуру безпеки» організації, а не лише кількість кліків.
- Необхідна інформація: Надає спеціалізовані модулі для стандартів відповідності, таких як HIPAA, PCI DSS та GDPR.

PhishGrid

Найкраще для: Гнучке моделювання та навчання на основі API. PhishGrid зосереджується на перетині симуляції та інтерактивного навчання. Він розроблений для аналізу зрілості середовища та пропонує більш модульний підхід до навчання, який часто використовують постачальники послуг.

Чому це працює:

- Моделювання в реальному часі: підкреслює безпосередність циклів зворотного зв'язку під час процесу моделювання.
- Комплексна звітність: Надає детальну аналітику, яка дозволяє командам детально виявляти вразливості конкретних відділів.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети).

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною мовних властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython.

Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючою комою, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (*class*), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиначне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться *int*, *long*, *complex* і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. КORTEЖ у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх.

Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий.

Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформулювати висновки про виконаний обсяг робіт та одержані результати.

К6ПЗ-2026

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Фішинг зростає завдяки постійній адаптації, а не просто через обсяг повідомлень. Проста крадіжка облікових даних переросла у фінансове шахрайство, корпоративне проникнення та повномасштабну компрометацію ідентифікаційних даних.

Зростання цифрового банкінгу, платформ SaaS та віддаленої співпраці створило ширші поверхні для атак. Розслідування безпеки постійно виявляють фішинг як основну точку входу у великі інциденти порушення безпеки в різних галузях.

Фінансові збитки зростають разом із технічним удосконаленням, сягнувши мільярдів щорічних збитків у всьому світі. У 2026 році фішинг спрямований на системи автентифікації, токени сеансів та виконавчу владу, замість того, щоб покладатися лише на оманливі електронні листи.

Фішинг у 2026 році зосереджений на контролі ідентифікаційних даних, а не на простому викраденні облікових даних. Зловмисники прагнуть отримати доступ до облікового запису, токени сеансу та привілейовану авторизацію замість того, щоб просто збирати паролі.

Автоматизація трансформувала проведення кампаній, дозволивши зловмисникам персоналізувати повідомлення у великих масштабах. Викрадені дані, публічні профілі та зламані бази даних сприяють дуже переконливим спробам видавати себе за іншу особу.

Контроль безпеки тепер стикається з тиском на рівні автентифікації, де основними цілями стають push-схвалення, дозволи OAuth та сеансові файли cookie. Сучасні фішингові кампанії поєднують соціальну інженерію, технічну експлуатацію та маніпуляції інфраструктурою в одному ланцюжку атак.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

Які методи фішингу спрямовані на людину?

Фішинг, спрямований на людей, спирається на психологічні маніпуляції, а не виключно на технічну експлуатацію. Довіра, терміновість, авторитет та знайомство залишаються основними рушійними силами успішної компрометації.

1. Фішинг з використанням копії

Фішинг-атака спрямована на конкретну особу з використанням персоналізованих даних, зібраних із публічних профілів, витоків даних або корпоративних джерел. Повідомлення точно відображають легітимне ділове спілкування, що ускладнює виявлення без перевірки.

2. Напади китобійного промислу

Китобійний вилов зосереджений на керівниках та особах вищої ланки, які приймають рішення з фінансовою владою. Зловмисники розробляють сценарії з високими ставками, включаючи юридичні повідомлення, придбання або термінові банківські перекази.

3. Посміхаючись

Smishing розсилає шкідливі посилання через текстові повідомлення, які виглядають так, ніби надходять від банків, служб доставки або систем автентифікації. Мобільні інтерфейси обмежують видимість URL-адрес, що збільшує ймовірність випадкових кліків.

4. Вішинг

Вішинг використовує телефонні дзвінки, щоб видавати себе за фінансові установи, внутрішні IT-команди або державні установи. Інструменти маніпулювання голосом підвищують реалізм і зменшують підозри під час живої взаємодії.

5. Фішинг у соціальних мережах

Фішинг у соціальних мережах поширюється через особисті повідомлення, фальшиві пропозиції роботи або видавані акаунти брендів. Жертв перенаправляють на сторінки збору облікових даних, які імітують перевірені платформи.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

Фішинг за допомогою штучного інтелекту підвищує реалізм, масштаб та персоналізацію на кожній поверхні атаки. Автоматизована генерація контенту дозволяє зловмисникам відтворювати стиль письма, тон та контекстуальну релевантність з мінімальними зусиллями.

6. Фішинг електронною поштою, згенерований штучним інтелектом

Фішингові електронні листи, згенеровані штучним інтелектом, відтворюють корпоративні мовні моделі та стилі внутрішнього спілкування. Повідомлення містять контекстуальні посилання, які зменшують підозри та підвищують рівень реагування.

7. Фішинг з використанням глибоких фейкових голосових атак

Голосовий фішинг Deepfake використовує моделі синтезованого мовлення, навчені на коротких публічних аудіозразках. Шахраї видають себе за керівників або довірених осіб під час складних фінансових запитів.

8. Фішинг за допомогою чат-ботів

Фішинг на основі чат-ботів розгортає автоматизовані агенти для спілкування на шахрайських веб-сайтах. Жертви взаємодіють у режимі реального часу та розкривають облікові дані або платіжні дані, не розпізнаючи обман.

9. Фішинг-клонування на базі штучного інтелекту

Фішинг-клонування на основі штучного інтелекту дублює легітимні електронні листи та замінює справжні посилання шкідливими. Інструменти автоматизації прискорюють реплікацію між кількома цілями за лічені хвилини.

Фішинг на основі автентифікації спрямований на системи перевірки особи, а не лише на поштові скриньки. Зловмисники зосереджуються на отриманні схваленого доступу шляхом зловживання згодою, крадіжки токенів та маніпуляції push-сповіщеннями.

10. Напади втоми від MFA

Атаки типу «втома MFA» надсилають повторювані push-сповіщення на пристрій цілі, доки один запит не буде схвалено. Постійні запити створюють плутанину та збільшують ймовірність випадкової авторизації.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

11. Фішинг за допомогою згоди OAuth

Фішинг за допомогою OAuth дозволяє користувачам обманом надати шкідливим програмам доступ до своїх облікових записів. Схвалені дозволи забезпечують постійний доступ без необхідності крадіжки пароля.

12. Атаки типу «браузер у браузері»

Атаки типу «браузер у браузері» відображають реалістичні спливаючі вікна для входу на шкідливих сторінках. Жертви вводять облікові дані у вікно імітації автентифікації, яке виглядає як законне.

13. Фішинг із викраденням сеансів

Викрадення сеансу захоплює файли cookie автентифікації після успішного входу. Зловмисники повторно використовують токени сеансу, щоб обійти контроль пароля та багатфакторної перевірки.

Фішинг на рівні інфраструктури маніпулює каналами розповсюдження, а не окремими поштовими скриньками. Зловмисники ставлять під загрозу видимість, рейтинги в пошуку, відносини з постачальниками та надійні механізми доставки, щоб розширити охоплення.

14. SEO-отруєння

SEO-отруєння розміщує шкідливі веб-сайти на вершині результатів пошуку за популярними запитам. Користувачі, які шукають портали входу або завантаження програмного забезпечення, потрапляють на шахрайські сторінки, які збирають облікові дані.

15. Фішинг QR-кодів

Фішинг за допомогою QR-кодів вбудовує шкідливі посилання в коди, що скануються, розміщені в електронних листах, рахунках-фактурах, паркоматах або публічних плакатах. Мобільне сканування зменшує видимість повної URL-адреси та підвищує довіру до взаємодії.

16. Фішинг ланцюга поставок

Фішинг через ланцюги поставок змушує постачальників, підрядників або сторонніх постачальників послуг поширювати шкідливі посилання через надійні канали зв'язку. Внутрішні команди швидко реагують на запити знайомих партнерів та несвідомо розкривають облікові дані.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

3.2 Розробка структурної схеми

Цифрова ідентифікація тепер контролює доступ до банківських систем, SaaS-платформ, медичних записів та корпоративних мереж. Один скомпрометований обліковий запис часто забезпечує доступ до кількох підключених сервісів.

Системи автентифікації значною мірою залежать від схвалення push-повідомлень, сеансів браузера та делегованих дозволів. Зловмисники використовують людську втому, довіру до брендovаних інтерфейсів та надмірну впевненість у багатofакторному захисті.

Віддалена робота, мобільний доступ та постійне підключення до Інтернету збільшують ризик отримання оманливих запитів на різних пристроях. Великий обсяг повідомлень у поєднанні з реалістичною імітацією іншої особи зменшує вагання та пришвидшує реакцію жертви.

Сучасний захист від фішингу вимагає багаторівневих засобів контролю безпеки, орієнтованих на ідентифікацію, замість того, щоб покладатися лише на фільтрацію електронної пошти.

Багатofакторна автентифікація (MFA), стійка до фішингу

Апаратні ключі безпеки та автентифікація на основі ключів доступу блокують втому від схвалення та атаки повторного відтворення токенів. Верифікація, прив'язана до пристрою, значно зменшує спроби несанкціонованого доступу.

Нульова довіра

Безпека Zero Trust забезпечує постійну перевірку користувачів, пристроїв та сеансів. Рішення щодо доступу залежать від ідентифікаційних даних, стану пристроїв та сигналів поведінкового ризику.

Безпека електронної пошти

Розширені шлюзи електронної пошти аналізують поведінку відправника, контекст повідомлення та вбудовані посилання в режимі реального часу.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

Перевірка «пісочниці» запобігає потраплянню шкідливих вкладень до поштових скриньок співробітників.

Захист домену

Застосування DMARC, SPF та DKIM запобігає підміні електронної пошти та видаванню себе за інший бренд. Суворі політики домену зменшують зовнішнє зловживання корпоративною ідентичністю.

Навчання з безпеки

Регулярні симуляції фішингу покращують розпізнавання співробітників та точність звітності. Культура, що орієнтована на перевірку, знижує рівень успішності атак соціальної інженерії.

Як люди можуть захистити себе від розширеного фішингу?

Особиста безпека залежить від звичок перевірки та надійних практик захисту особистих даних.

Перевірка URL-адреси

Вручну перевіряйте адреси веб-сайтів, перш ніж вводити облікові дані або платіжні дані. Звертайте увагу на незначні орфографічні зміни, зайві символи або незнайомі домени.

Гігієна паролів

Використовуйте менеджер паролів для створення та зберігання унікальних облікових даних для кожного облікового запису. Унікальні паролі запобігають повторному використанню зловмисниками викрадених даних для входу на різних платформах.

Сильний багатосторонній фінансовий план

Увімкніть автентифікацію на основі додатків або апаратні ключі безпеки для критично важливих облікових записів. Фізична або пов'язана з пристроєм перевірка блокує більшість спроб віддаленого захоплення доступу.

Обережно з QR-кодом

Уникайте сканування QR-кодів з невідомих електронних листів, плакатів або рахунків-фактур. Шкідливі коди часто перенаправляють на фальшиві портали входу, оптимізовані для екранів мобільних пристроїв.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

Моніторинг облікового запису

Регулярно перевіряйте банківські виписки, активність входу в систему та сповіщення безпеки. Раннє виявлення обмежує фінансові збитки та тривалість несанкціонованого доступу.

Оновлення пристроїв

Негайно встановлюйте оновлення операційної системи та програм на всіх пристроях. Патчі безпеки усувають вразливості, які часто використовують фішингові пакети.

Перевірка електронної пошти

Ставтеся до неочікуваних вкладень та термінових запитів зі скептицизмом, навіть від знайомих контактів. Пряме підтвердження через офіційні канали зменшує ризик видавання себе за іншу особу.

На що слід звернути увагу в антифішинговому рішенні безпеки у 2026 році?

Ефективний захист від фішингу вимагає ідентифікаційної аналітики, моніторингу поведінки та можливостей автоматичного реагування.

Виявлення ШІ

Платформи безпеки повинні аналізувати поведінкові аномалії, а не покладатися лише на зіставлення сигнатур. Моделі машинного навчання покращують виявлення нових фішингових моделей.

Захист токенів

Рішення повинні контролювати токени сеансу, дозволи OAuth та делеговані гранти доступу. Сповіщення в режимі реального часу зменшують вплив несанкціонованого контролю облікових записів.

Сканування в режимі реального часу

Перевірка посилання під час кліку запобігає перенаправленню на шкідливі домени. Динамічний аналіз блокує доставку корисного навантаження до введення облікових даних.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

Видимість SaaS

Хмарний моніторинг має відстежувати поведінку під час входу в систему, неможливі моделі подорожей та аномальну активність API. Централізовані панелі інструментів покращують швидкість реагування в розподілених середовищах.

Автоматизована відповідь

Автоматизоване стримування ізолює скомпрометовані облікові записи одразу після появи підозрілої активності. Швидке реагування обмежує горизонтальне переміщення та витік даних.

Розвідка про загрози

Інтегровані канали загроз посилюють виявлення нових фішингових доменів та інфраструктури. Постійні оновлення підтримують актуальність проти тактик, що розвиваються.

Фішинг у 2026 році функціонує як структурована стратегія атаки на ідентифікаційні дані, а не як просте електронне шахрайство. Людська психологія, системи автентифікації та цифрова інфраструктура тепер служать скоординованими точками входу для компрометації.

Захист вимагає багаторівневого захисту для людей, процесів і технологій. Організації та окремі особи, які надають пріоритет безпеці ідентифікаційних даних, дисципліні перевірки та постійному моніторингу, значно зменшують ризик сучасних фішингових загроз.

На структурній схемі (рисунок 3.1) зображена розроблена під час дипломного проектування система реалізації методів соціального інжинірингу через адресу e-mail.

Firefox розповсюджується із частково відкритим кодом і має розширений набір засобів (Software Development Kit) для написання й розповсюдження додатків на його основі.

Як показано на рисунку 3.1, система реалізації методів соціального інжинірингу через адресу e-mail заснована на браузері Firefox і робить взаємодію через Software Development Kit.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

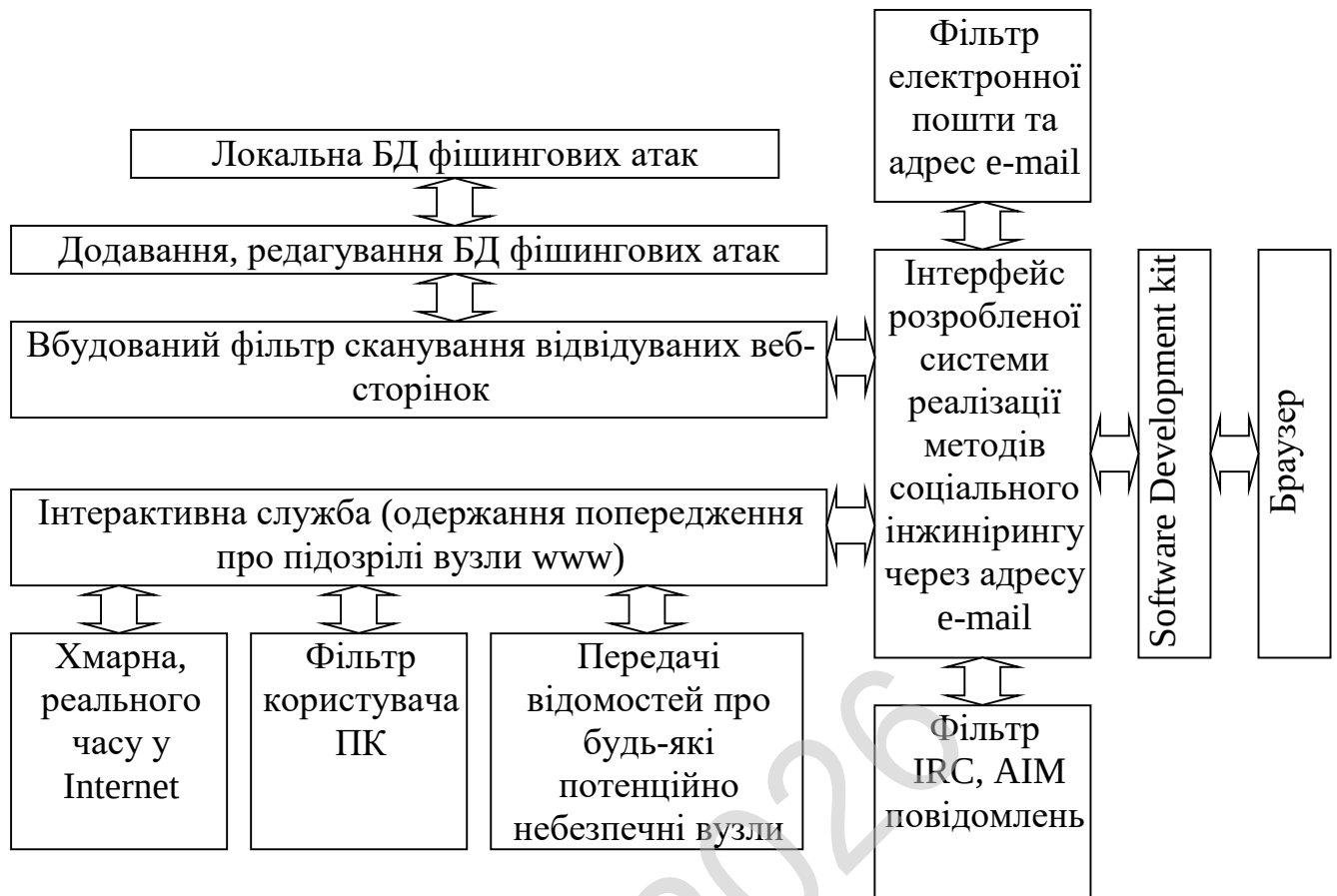


Рисунок 3.1 – Структурна схема системи

Розглянемо загальні можливості розробленого програмного забезпечення зображені на схемі. Через розроблену систему реалізації методів соціального інжинірингу через адресу e-mail відбуваються наступні дії:

1. Фільтр електронної пошти.
2. Фільтр IRC, AIM повідомлень.
3. Інтерактивна служба:

3.1. Реального часу на основі IE 7.0, Opera Software (GeoTrust) у глобальній мережі.

3.2 Фільтр користувача в ПК.

3.3 Передати відомості про будь-які потенційно небезпечні вузли.

4. Вбудований фільтр сканування відвідуваних веб-сторінок:

4.1 Додавання, Редагування БД.

4.2 Локальна БД фішингових атак.

За допомогою даних засобів імовірність фішингової атаки через електронну пошту й Sram, фішинг-атаки з використанням web-контента, фальсифіція рекламних банерів, IRC і передача ІМ-повідомлень, використання троянських програм значно зменшується надаючи користувачеві надійну систему захисту.

3.3 Розробка функціональної схеми

На функціональній схемі, зображеній на рисунку 3.2, є можливість прослідкувати за шляхами проходження функціональних сигналів від одного функціонального блоку до іншого та побачити рівні надходження даних з мережі та міри їхньої обробки.

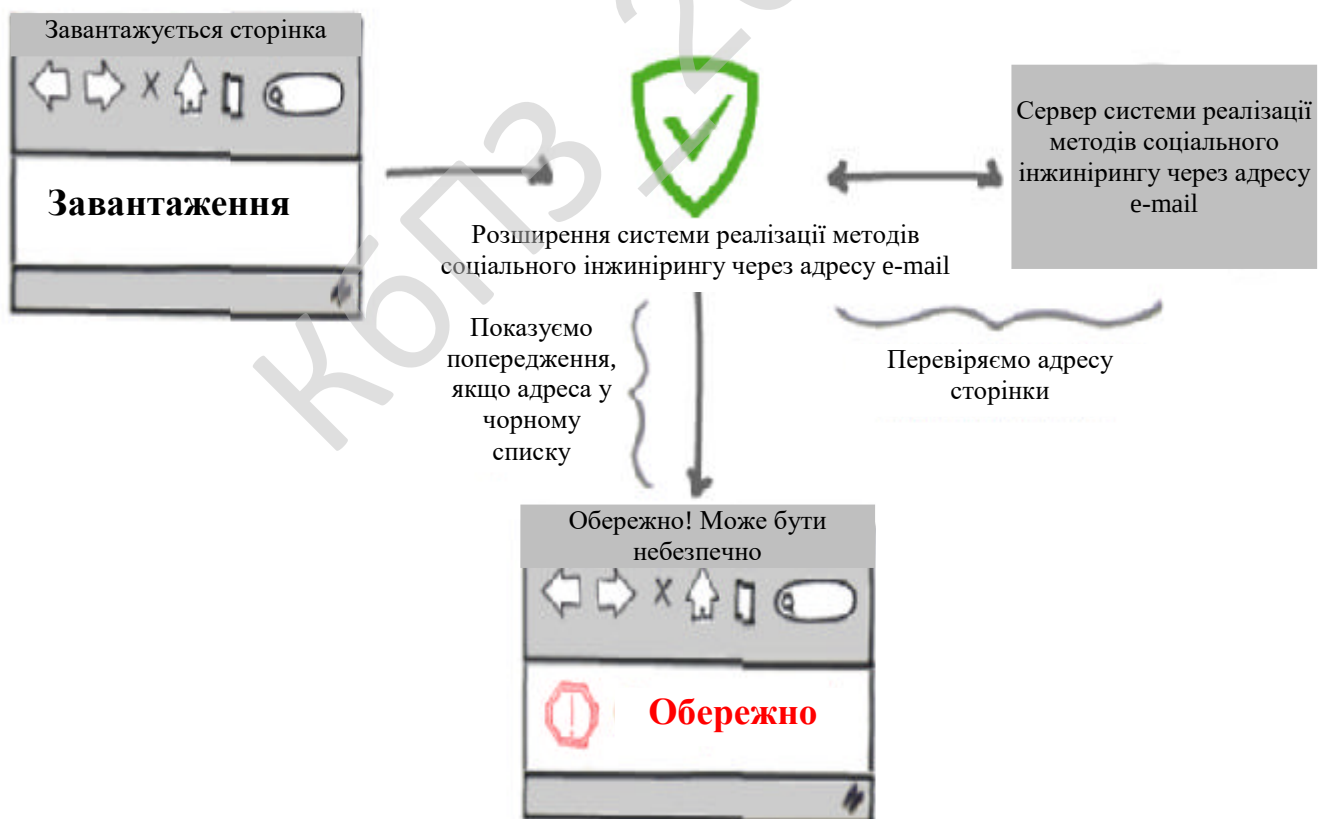


Рисунок 3.2 – Функціональна схема системи

Користувач використовуючи Інтернет браузер FireFox одержує доступ до глобальної мережі Інтернет та через розроблене програмне забезпечення взаємодіє з Інтернетом (Web-контент), розмовляє за допомогою Інтернет пейджерів (IRC, ICQ, AIM), користується електронною поштою.

Firefox є релізом наступного покоління веб-браузерів, який має безліч нагород, він містить у собі безліч спеціальних можливостей, що дозволяють зробити веб-браузер і веб-контент доступним для всіх користувачів.

Програмне забезпечення є системою розширення (extentions) можливостей FireFox надаючи користувачеві захист від фішингових атак.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходить взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

- Сховища даних (репозиторії).
- Зовнішні по відношенню до системи сутності.
- Потоки даних між елементами трьох попередніх типів.

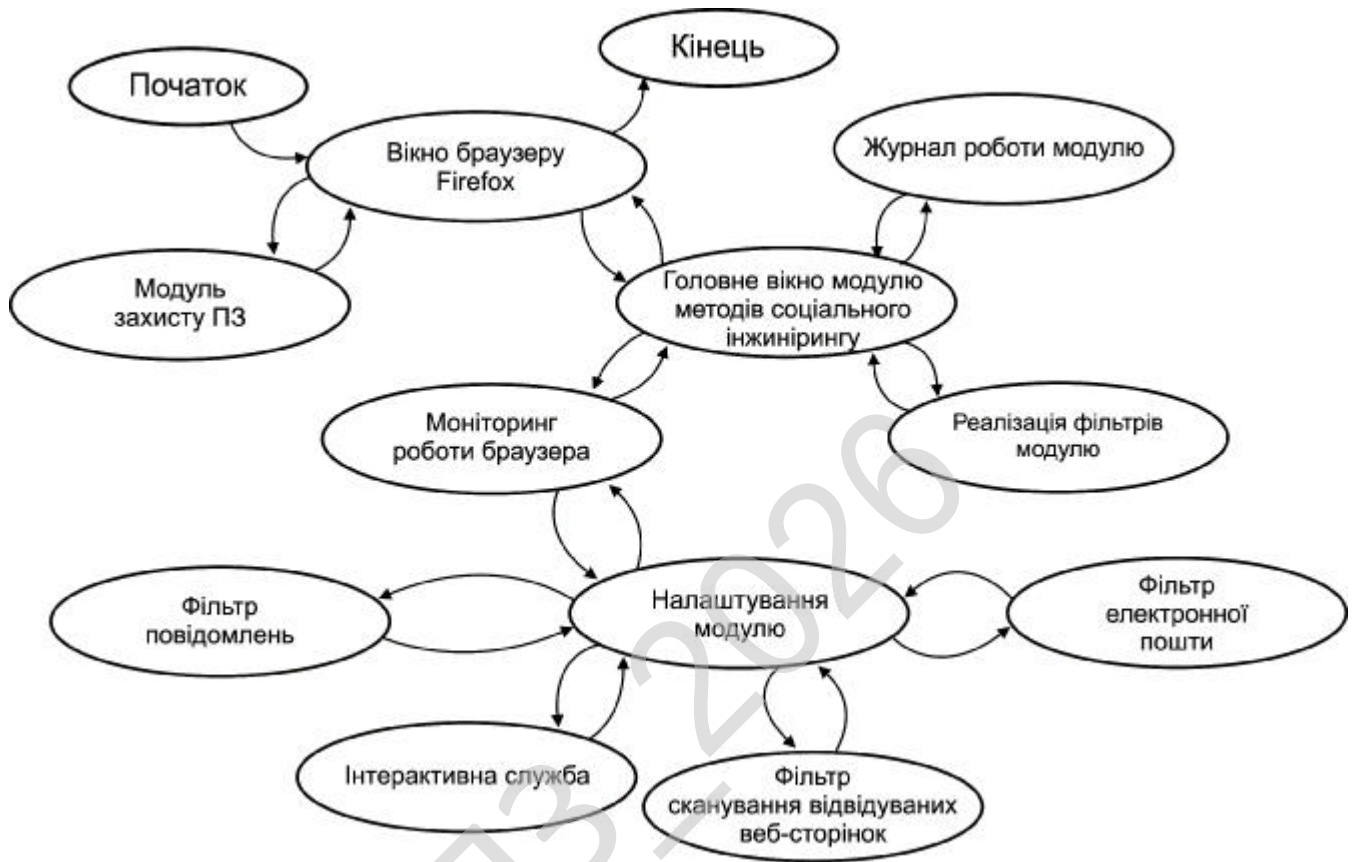


Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile.

Гнучка розробка програмного забезпечення (Agile software development, agile-методи) – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

Гнучка розробка – найкращий засіб для підвищення продуктивності розробників програмного забезпечення.

Більшість гнучких методологій націлені на мінімізацію ризиків, шляхом зведення розробки до серії коротких циклів, що мають назву ітерацій, які зазвичай тривають один-два тижні. Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі, і включає всі завдання, необхідні для видачі мінімального приросту за функціональністю: планування, аналіз вимог, проектування, кодування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі те, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації, команда виконує переоцінку пріоритетів розробки.

Agile акцентує увагу на безпосередньому спілкуванні «віч-на-віч». Більшість agile команд розташовані в одному офісі, його іноді називають bullpen. Як мінімум вона включає і «замовників» (замовники, які визначають продукт, також це можуть бути менеджери продукту, бізнес аналітики або клієнти). Офіс

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

може також включати тестувальників, дизайнерів інтерфейсу, технічних авторів і менеджерів.

Основною метрикою agile методів є робочий продукт. Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації в порівнянні з іншими методами. Це привело до критики цих методів як недисциплінованих.

Agile – родина процесів розробки, а не єдиний підхід в розробці програмного забезпечення, і визначається Agile Manifesto. Agile не включає практик, а визначає цінності та принципи, якими керуються успішні команди.

Agile Manifesto розроблений і прийнятий 17 розробниками 11-13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Маніфест підписали представники наступних методологій Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature driven development, Pragmatic Programming. Agile Manifesto містить 4 основні ідеї та 12 принципів. Примітно, що Agile Manifesto не містить практичних порад.

Основні ідеї:

- Особистості та їхні взаємодії важливіші, ніж процеси та інструменти.
- Робоче програмне забезпечення важливіше, ніж повна документація.
- Співпраця із замовником важливіша, ніж контрактні зобов'язання.
- Реакція на зміни важливіша, ніж дотримання плану.

Принципи, які роз'яснює Agile Manifesto:

- задоволення клієнта за рахунок ранньої та безперебійної поставки коштовного програмного забезпечення;
- вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);
- часта поставка робочого програмного забезпечення (кожен місяць або тиждень або ще частіше);
- тісне, щоденне спілкування замовника з розробниками впродовж всього проекту;

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		31

- проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;
- рекомендований метод передачі інформації – особиста розмова (віч-на-віч);
- робоче програмне забезпечення – найкращий вимірник прогресу;
- спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;
- постійну увагу поліпшенню технічної майстерності та зручному дизайну;
- простота – мистецтво не робити зайвої роботи;
- найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованої команди;
- постійна адаптація до мінливих обставин.

Маніфест та Принципи гнучкої розробки містять високорівневі ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти й створювати команди, в яких приємно та цікаво працювати.

Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити. По-іншому й не могло бути, оскільки Маніфест та Принципи народилися внаслідок консенсусу представників різних (хоча й споріднених) напрямів, які могли знайти спільну основу лише на рівні базових цінностей та принципів.

Критика. Багато керівників проектів, що працюють у традиційних методологіях на кшталт «водоспаду», критикують agile-методи.

Один з повторюваних пунктів критики: при agile-підході часто нехтують створенням «дорожньої карти» розвитку продукту, так само як і управлінням вимогами, в процесі якого і формується така «карта». Гнучкий підхід до управління вимогами не має на увазі далекосяжних планів (по суті, управління вимогами просто не існує в даній методології), а має на увазі можливість замовника раптом і несподівано наприкінці кожної ітерації виставляти нові

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

вимоги, що часто суперечать архітектурі вже створеного і поставленого продукту. Таке іноді призводить до катастрофічних «авралів» з масовим рефакторингом і переробками практично на кожній черговій ітерації.

Крім того вважається, що робота в agile мотивує розробників вирішувати всі прибулі завдання найпростішим і найшвидшим можливим способом, при цьому часто не звертаючи уваги на коректність коду з точки зору вимог базової платформи (підхід «працює, та й добре», при цьому не враховується, що може перестати працювати при найменшій зміні або ж породити важкі до відтворення дефекти після реального розгортання у клієнта). Це призводить до зниження якості продукту і накопиченню дефектів.

Методології. Існують методології, які дотримуються цінностей і принципів заявлених в Agile Manifesto, деякі з них:

1. Agile Modeling – набір понять, принципів і прийомів (практик), що дозволяють швидко і просто виконувати моделювання і документування в проектах розробки програмного забезпечення. Не включає в себе детальну інструкцію з проектування, не містить описів, як будувати діаграми на UML.

Основна мета – ефективне моделювання і документування; але не охоплює програмування та тестування, не включає питання управління проектом, розгортання і супроводу системи. Однак включає в себе перевірку моделі кодом.

2. Agile Unified Process (AUP) спрощена версія IBM Rational Unified Process (RUP), розроблена Скоттом Амблером, яка описує просте і зрозуміле наближення (модель) для створення програмного забезпечення для бізнес-додатків.

3 Agile Data Method – група ітеративних методів розробки програмного забезпечення, в яких вимоги та рішення досягаються в рамках співпраці різних крос-функціональних команд.

4. DSDM заснований на концепції швидкої розробки додатків (Rapid Application Development, RAD). Являє собою ітеративний і інкрементний підхід,

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

який надає особливого значення тривалій участі в процесі користувача/споживача.

5. Essential Unified Process (EssUP).

6. Екстремальне програмування (Extreme programming, XP).

7. Feature driven development (FDD) – функціонально-орієнтована розробка.

Використовуване в FDD поняття функції або властивості (feature) Системи досить близько до поняття прецеденту використання, використовуваному в RUP, істотна відмінність – це додаткове обмеження: «кожна функція повинна допускати реалізацію не більше, ніж за два тижні». Тобто якщо сценарій використання досить малий, його можна вважати функцією. Якщо ж великий, то його треба розбити на декілька відносно незалежних функцій.

8. Getting Real – ітераційний підхід без функціональних специфікацій, що використовується для веб-додатків. У даному методі спершу розробляється інтерфейс програми, а потім її функціональна частина.

9. OpenUP – це ітераційно-інкрементний метод розробки програмного забезпечення. Позиціюється, як легкий і гнучкий варіант RUP. OpenUP ділить життєвий цикл проекту на чотири фази: початкова фаза, фази уточнення, конструювання та передачі. Життєвий цикл проекту забезпечує надання зацікавленим особам та членам колективу точок ознайомлення і прийняття рішень впродовж усього проекту. Це дозволяє ефективно контролювати ситуацію і вчасно приймати рішення про задовільність результатів. План проекту визначає життєвий цикл, а кінцевим результатом є остаточний додаток.

10. Scrum встановлює правила керування процесом розробки та дозволяє використовувати вже існуючі практики кодування, коректуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на більш ранніх етапах розробки програмного продукту.

11. Бережлива розробка програмного забезпечення (lean software development). Використовує підходи з концепції бережливого виробництва.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		34

Було використано підхід на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінгхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		35

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В ХР ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

- Спроба примирити гуманність і продуктивність.
- Механізм для соціальної зміни.
- Шлях до удосконалення.
- Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

ХР використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		36

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

ХР намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн.

Написання коду. Прихильники ХР заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми ХР. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).
 - 1.1. Розробка через тестування (Test driven development).
 - 1.2 Гра в планування (Planning game).
 - 1.3. Замовник завжди поруч (Whole team, Onsite customer).
 - 1.4 Парне програмування (Pair programming).
2. Безперервний, а не пакетний процес.
 - 2.1 Безперервна інтеграція (Continuous Integration).
 - 2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4 Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних - прямокутником, блок прийняття рішень - ромбом, еліпсом - початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування - початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи реалізації методів соціального інжинірингу, модулю обробки помилок програми і основному модулю.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

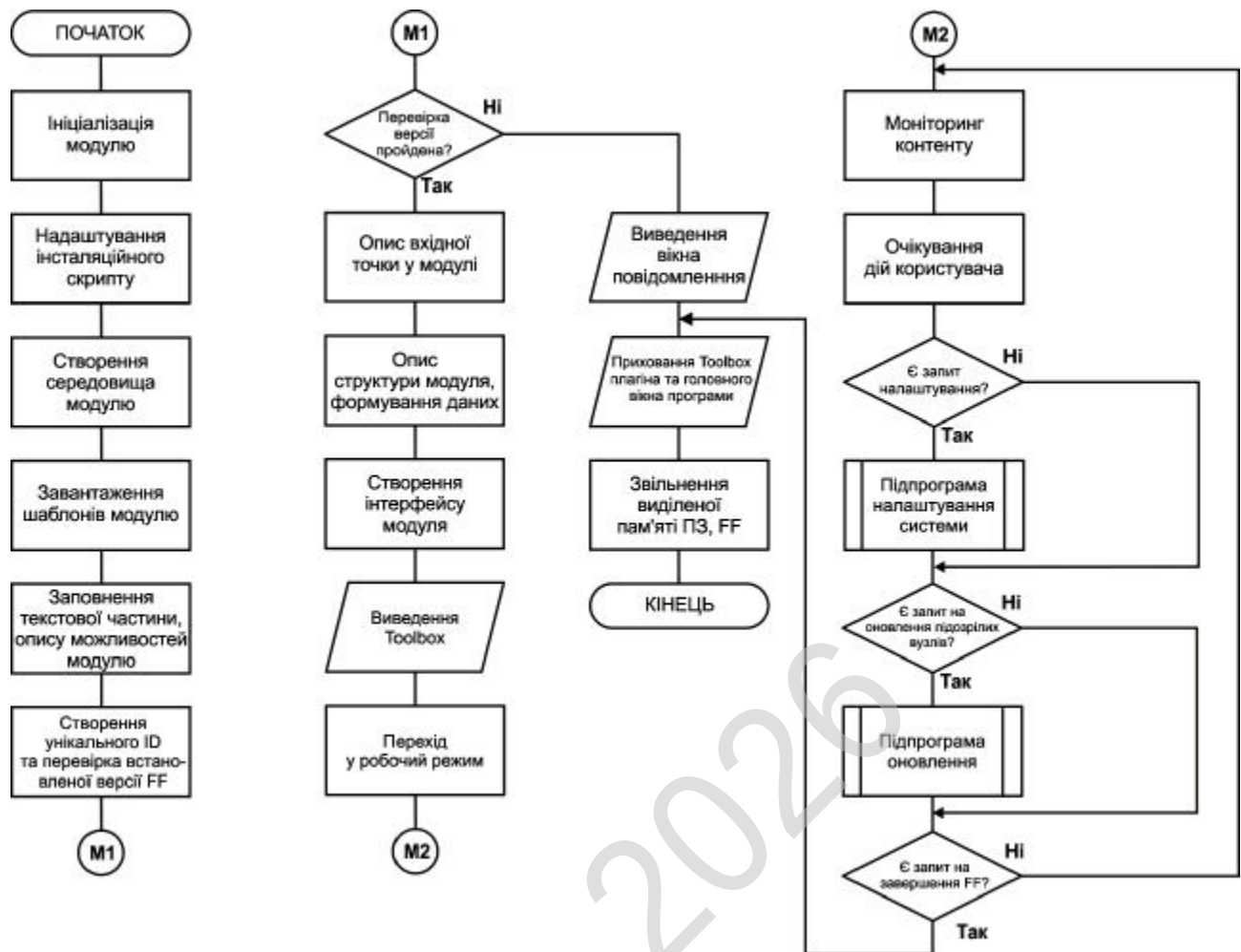


Рисунок 4.1 – Блок-схема основної програми

Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

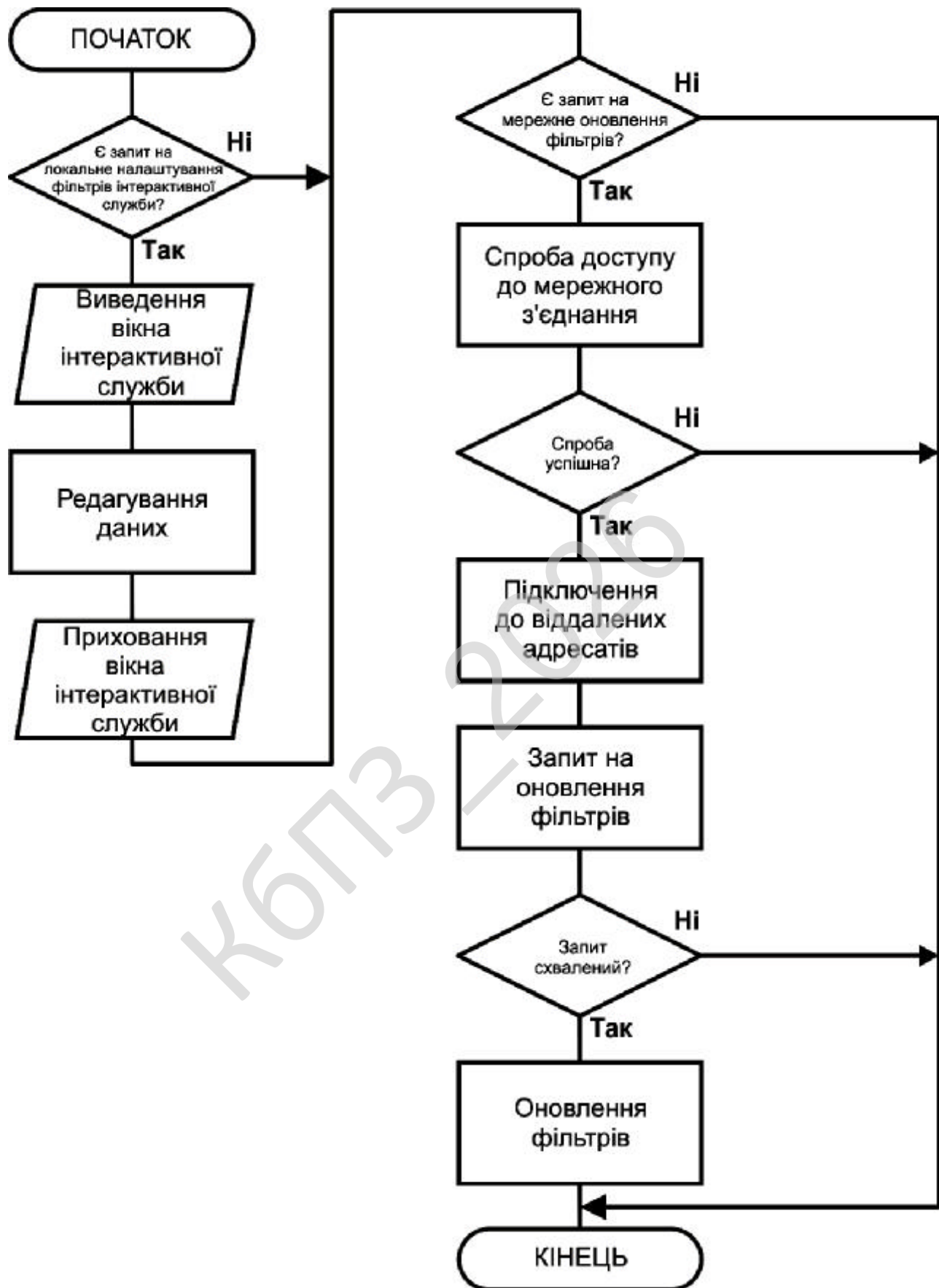


Рисунок 4.2 – Блок-схема роботи підпрограми

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

- Керівникам проектів, які керують розподілом завдань і контролем за проектом.
- Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.
- Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.
- Програмістам які реалізують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розширення FireFox мають власну структуру файлів і щоб розширення правильно працювало, потрібно створити кореневий каталог, де будуть розміщатися всі файли розширення антифішингового захисту.

Створюються вкладені шаблонів директорії. Їхні імена повинні бути chrome (обов'язково в нижньому регістрі) і content (знову в нижньому регістрі). Таким чином, створюється файлова структура наступного виду:

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		43

+ BR/
+- chrome/
+- content/

У будь-якому розширенні FireFox повинне бути як мінімум два файли – install.rdf і contents.rdf. У першому описано, як розширення буде встановлюватися в браузер, другий файл визначає структуру розширення.

Це прості текстові файли в XML-форматі, так що їх можна створювати в будь-якому текстовому редакторі. Але для зручності, я користувався спеціальним редактором для xml-файлів (XMLSpy). Розширення rdf – це скорочення для "Resource Description Framework".

Вміст файлу *install.rdf* – цей файл повинен розташовуватися на самому верхньому рівні нашої антифішингової файлової ієрархії.

+ BR /
+- install.rdf
+- chrome/
+- content/

У цьому файлі було описане наше розширення для менеджера розширень браузера FireFox, повідомивши його різну корисну інформацію- з якими версіями браузера сумісний додаток BR.

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"xmlns:em="http://www.mozilla.org/2004/em-rdf#">
<Description about="urn:mozilla:install-manifest">
<em:description>BR extention</em:description>
<em:id>{65b3130e-8513-41b6-8ea8-43dbd9cc0b12}</em:id>
<em:name>BR</em:name>
<em:version>1.0</em:version>
<em:targetApplication>
<Description>
<em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
<em:minVersion>1.0</em:minVersion>
<em:maxVersion>1.0</em:maxVersion>
</Description>
</em:targetApplication>
<em:file>
<Description about="urn:mozilla:extension:file: BR.jar">
<em:package>content/</em:package>
```

```
</Description>
</em:file>
```

```
</Description>
</RDF>
```

У перших рядках написано, що цей файл – ніщо інше, як стандартний XML-Файл. Для менеджера розширень надаються наступні дані.

```
<em:description>BR extention</em:description>
<em:id>{65b3130e-8513-41b6-8ea8-43dbd9cc0b12}</em:id>
<em:name>BR</em:name>
<em:version>1.0</em:version>>
```

Перший рядок указує на авторські права коду, для чого використовується тег <em:creator>. Наступний рядок – короткий опис того, а для чого, властиво, це розширення призначене. Ця інформація буде видна користувачам у менеджері розширень. Третій рядок указує на домашню сторінку розширення.

Четвертий рядок у цій секції, у тезі <em:id>, це найбільш важлива інформація у всьому файлі. Тут вказується унікальний ідентифікатор антифішингового розширення. Існує багато способів генерації унікальних ідентифікаторів.

Я використовував програму GUID Generator <http://www.hoskinson.net/webservices/guidgeneratorclient.aspx>.

Після створення унікального ідентифікатора я помістив його в тег <em:id>. У наступних рядках файлу описується, для якого додатка призначене це розширення:

```
<em:targetApplication>
  <Description>
    <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
    <em:minVersion>1.0</em:minVersion>
    <em:maxVersion>1.0</em:maxVersion>
  </Description>
</em:targetApplication>
```

У моєму випадку в тезі <em:name> зазначений унікальний код браузера FireFox і його значення не повинне змінюватися. Тут же я встановив два тега для

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

вказівки мінімальної й максимальної версії додатка, з якими дане розширення сумісне.

У моєму випадку ці два значення збігаються. Поточна версія FireFox дотримується наступної угоди по версіях:

```
major.minor.release.build[+]
```

(Поля, що впливають за *major* – необов'язкові)

У кінцевій частині опису файлу я вказав розташування JAR-Файлу (розглянутий нижче):

```
<em:file>
<Description about="urn:mozilla:extension:file: BR.jar">
  <em:package>content/</em:package>
</Description>
</em:file>
```

В атрибуті *about* тегу <Description> вказується ім'я. У моєму випадку це посилання на файл "BR.jar".

Для зручності, я назвав ім'я файлу співпадаюче з ім'ям розширення. В елементі <em:package> вказується розташування JAR-Файлу. Тому що в нас у директорії chrome є тільки один фолдер content, те його він і вказується.

Уміст contents.rdf – у цьому файлі описується, як моє розширення зберігається.

Це текстовий файл в XML-Форматі й він повинен розташовуватися в директорії content. Таким чином, після створення цього файлу, я одержав наступну структуру:

```
+ - BR/
  +- install.rdf
  +- chrome/
    +- content/
      +- contents.rdf
```

Зміст цього файлу наступний:

```
<?xml version="1.0"?>
<RDF:RDF xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:chrome="http://www.mozilla.org/rdf/chrome#">
<RDF:Seq about="urn:mozilla:package:root">
<RDF:li resource="urn:mozilla:package:BR"/>
```

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		46

```

</RDF:Seq>
<RDF:Description about="urn:mozilla:package:BR"
chrome:extension="true" chrome:name="BR"/>
<RDF:Seq about="urn:mozilla:overlays">
<RDF:li resource="chrome://browser/content/browser.xul"/>
</RDF:Seq>
<RDF:Seq about="chrome://browser/content/browser.xul">
<RDF:li>chrome://BR/content/BR-Overlay.xul</RDF:li>
</RDF:Seq>
</RDF:RDF>

```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки підключення шифрування. Половина n-бітного ключа шифрування використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

- 128-бітний блочний симетричний шифр.
- Довжина ключів 128, 192 і 256 біт.
- Відсутність слабких ключів.
- Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamard Transform, PHT).

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного хог. Це дає можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening),

проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура хог'а даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3]. Розробники Twofish стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного відбілювання, оскільки вона безпосередньо використовує хог підключів з вхідними даними. У результаті дослідники показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x_0, x_1, x_2, x_3 . Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4×4 , причому обчислення проводяться в скінченному полі по модулю

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

непривідного многочлена. MDS матриця – це така матриця над кінцевим полем K , що якщо взяти її як матрицю лінійного перетворення з простору U у простір V , то будь-які два вектори з простору U виду $(x, f(x))$ будуть мати як мінімум $m+1$ відмінностей в компонентах. Тобто набір векторів вигляду $(x, f(x))$ утворює код, що володіє властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона. В Twofish властивість максимальної рознесеності матриці MDS означає, що загальна кількість змінних байт вектора a і вектора b не менше п'яти. Іншими словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b .

Криптоперетворення Адамара (Pseudo-Hadamard Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною $2n$. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

$$\begin{pmatrix} a \\ b \end{pmatrix} \rightarrow \begin{pmatrix} a \oplus b \\ b \end{pmatrix}$$

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER). В Twofish це перетворення використовується при змішуванні результатів двох g -функцій ($n = 32$).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які хог-яться з результатами функції F , додатково циклічно зрушуються на один біт. Третій блок зрушується до операції хог, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те, що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e . Потім за допомогою блоку M_o і функції h шифрується значення $2 * i$, а за допомогою блоку M_e шифрується значення $2*i+1$, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетворенням Адамара, і потім використовуються як раундові підключі.

КБПЗ – 2026

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи реалізації методів соціального інжинірингу через адресу e-mail. Для інсталяції розробленого ПЗ необхідно покроково виконати наступні дії – запустити Mozilla Firefox, через меню програми «Файл»-«Відкрити файл» запустити інсталяційний файл BR.xpi (файл розробленого модулю).

На рисунку 5.1 показано сторінку встановлення плагіну розробленого ПЗ у браузері Mozilla Firefox. Показано процес підтвердження встановлення. Браузер запитує дозволи, які потрібні для роботи плагіну. Розширення отримує доступ до вмісту сторінок електронної пошти, може аналізувати посилання, показувати сповіщення та зберігати службові налаштування. Після підтвердження з'являється повідомлення про успішне додавання ПЗ до Firefox.

Плагін позиціонується як засіб захисту від соціального інжинірингу через електронну пошту.

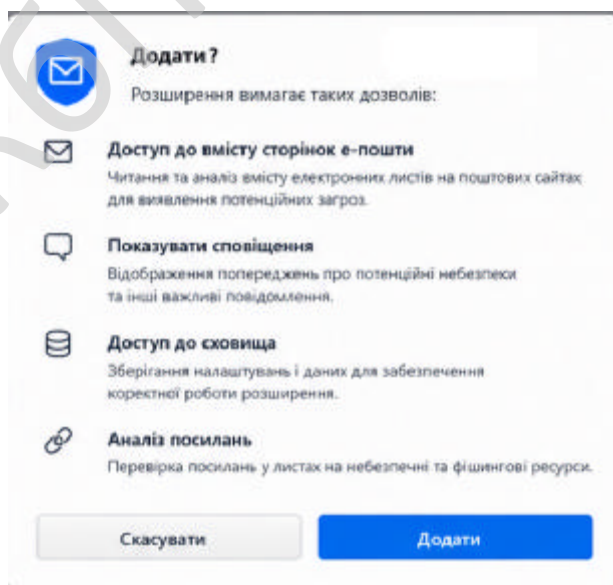


Рисунок 5.1 – Інсталяція розробленого ПЗ

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

На рисунку 5.2 показано роботу плагіну безпосередньо в поштовому сервісі. Користувач відкриває підозрілий лист, а розширення автоматично аналізує його вміст.

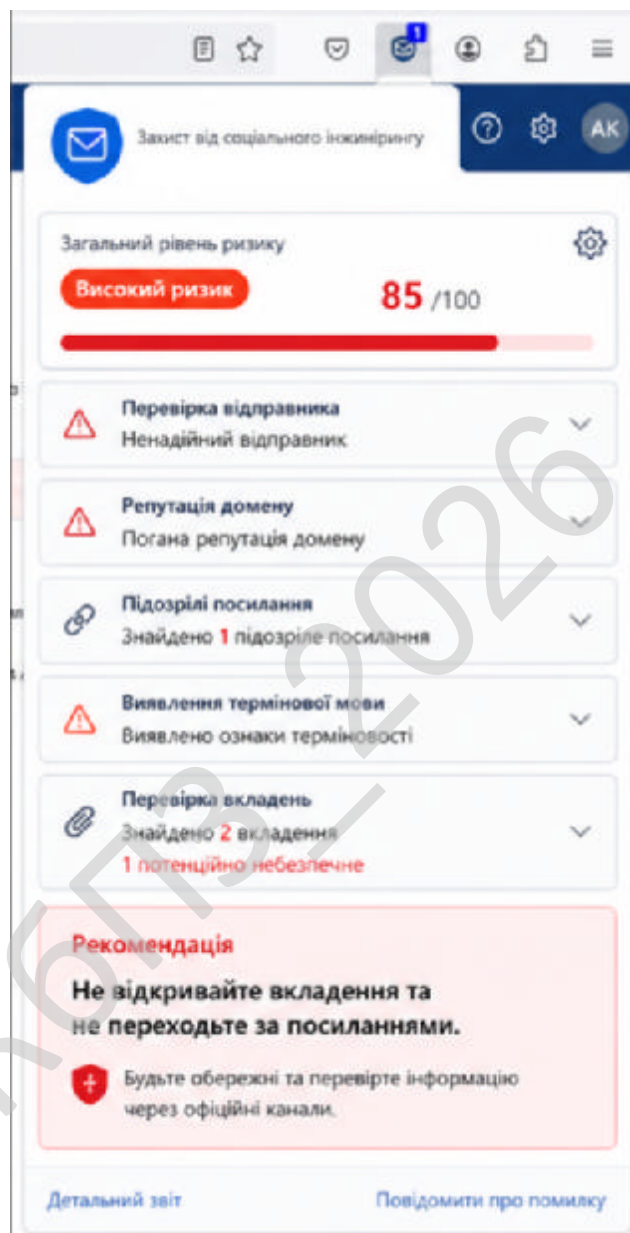


Рисунок 5.2 – Головне вікно ПЗ

Плагін визначає високий рівень ризику, перевіряє відправника, репутацію домену, наявність підозрілих посилань, ознаки термінової мови та вкладення. У результаті система показує попередження і рекомендує не відкривати вкладення

та не переходити за посиланнями. Також при переході до налаштувань відображається статистика перевірених листів, кількість заблокованих повідомлень, середній рівень ризику та стан активного захисту. Також є список останніх проаналізованих листів із зазначенням відправника, теми, статусу, рівня ризику та часу перевірки.

Основний функціонал плагіну охоплює автоматичну перевірку вхідних листів, виявлення фішингових повідомлень, аналіз доменів, перевірку вкладень, аналіз посилань, пошук ознак соціального інжинірингу, формування ризикового балу, блокування небезпечних листів, ведення журналу подій та створення звітів.

Додатково система підтримує білий і чорний списки відправників, ручний аналіз домену, експорт звіту, налаштування чутливості перевірки та перегляд рекомендацій щодо підвищення безпеки. Це дає змогу використовувати плагін як навчальний і практичний інструмент для протидії атакам через електронну пошту.

Система забезпечує контроль над всіма основними можливостями які надаються браузером і Інтернетом, а саме:

- захист електронної пошти;
- захист від фішинг-атак з використанням web-контента;
- захист від фальсифіції рекламних банерів;
- захист IRC, IRC, ICQ, AIM-повідомлень та завантаження троянських програм.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.3 зображено авторські дані розробленого програмного забезпечення.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		54

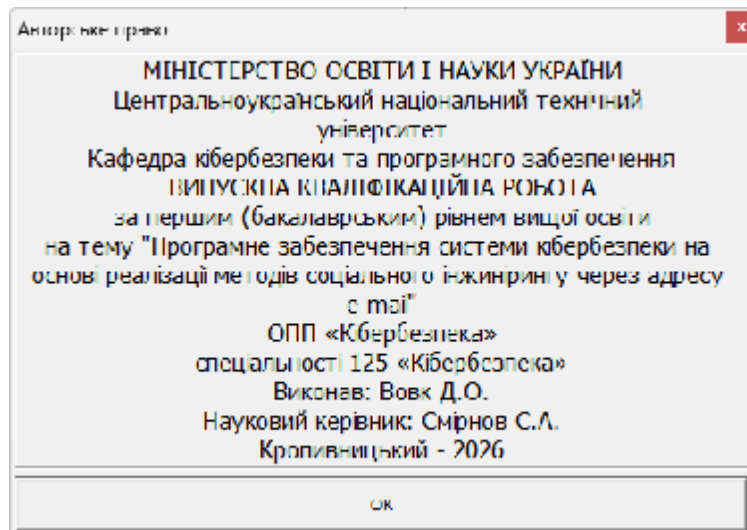


Рисунок 5.3 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

- Випуск.
- Встановлення та активація.
- Деактивація.

- Адаптація.
- Обновлення.
- Вмонтування.
- Відстежування версій.
- Видалення.
- Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження.

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		56

Проводилась оцінка:

- відповідності поставленим вимогам;
- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

- Як виконуються функції програми.
- Як приймаються вихідні дані.
- Як виробляються результати.
- Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^{10} . Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

Програмний виріб тут розглядається як «чорна скринька», чію поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

- Набір, утворений такими входними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

- Набір, утворений такими входними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

- Виявити такі входні дані, які з високою ймовірністю належать набору ІТс;
- Сформулювати такі очікувані результати, які з високою ймовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

- Некоректних чи відсутніх функцій;
- Помилки інтерфейсу;
- Помилки у зовнішніх структурах даних або в доступі до зовнішньої бази даних;
- Помилки характеристик (необхідна ємність пам'яті і т.д.);
- Помилки ініціалізації та завершення.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційних програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації, надають відкриті вихідні коди. Для

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		58

продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем на основі реалізації методів соціального інжинірингу через адресу e-mail.
- Досліджена система на основі реалізації методів соціального інжинірингу через адресу e-mail.
- На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання на основі реалізації методів соціального інжинірингу через адресу e-mail.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створенні програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

системи кібербезпеки на основі реалізації методів соціального інжинірингу через адресу e-mail. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p
2. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
3. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.
4. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.
5. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр “Кафедра”, 2025.– 320 с.
6. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка*. 2025. Том 1 № 29. С.704–716, 2025
7. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.
8. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

9. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

10. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

11. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings*, 2024, 3909, pp. 227–241.

12. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

13. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

14. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2024. № 2(26), С. 170–188.

15. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.

16. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка*. 2024. №3(23), С. 111-131.

17. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології*, 2024, № 13, с. 28-35.

18. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

19. Akhalaia, G., Iavich, M., Iashvili, G., Prysiashnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings*, 2023, 3550, pp. 313-320.

20. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

21. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

22. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings*, Volume 3504, 2023, pp. 1-11.

23. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebesko, B. «Selection of a Rational Composition of Information Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppapapati, C., Beligiannis, G.N. (eds) *Intelligent Communication Technologies and Virtual Mobile*

Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

24. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №3(19), 2023, С. 176-196.

25. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи розвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

26. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція “Інформаційна безпека та комп'ютерні технології” до 30-ти річчя кафедри кібербезпеки та програмного забезпечення*, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

27. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція “Інформаційна безпека та комп'ютерні технології” до 30-ти річчя кафедри кібербезпеки та програмного забезпечення*, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

28. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузових стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

29. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings*, Volume 3187, 2022,

					ВКРБ-125.26.0004.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		65

30. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

31. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління*, № 2(70). 2022. С. 28-37.

32. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 3(69). С. 93-98.

33. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»*, № 2 (307). С. 46-52. 2022.

34. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

35. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021*. P. 414-418

36. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 *IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

37. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». 2020 *IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020*, Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

38. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages 44-58.

39. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

40. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

41. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

42. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

43. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

44. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.
45. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.
46. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I., Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 166-171.
47. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 172-177.
48. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*, vol 48. Springer, Cham. 2021. pp 557-587.
49. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.
50. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

51. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.
52. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». *International Journal of Computing*; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.
53. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.
54. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.
55. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.
56. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019)*. 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.
57. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

58. Smirnov, O., Kuznetsov, A., Kiian, A., Pushkar'ov, A., Mialkovskiy, D., Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P. 707-712.

59. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

60. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

61. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019*, P. 395-399.

62. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019, Lviv, Ukraine, 2-6 July, 2019*, P. 129-134.